The Institution of Engineering and Technology

Why software is damaging the environment – and what can be done about it.

Tuesday, 22nd October 2024

Welcome

The lecture and webinar will start at 7:00PM BST

Introduction: George Williamson FIET Anglian Coastal Local Network

Presenter: Chris Watts Software Engineer and Technologist

Questions: via Q&A Messaging in the Teams session, and in person at the University of Suffolk. In Teams please type in your questions and these will be taken in a Q&A session at the end of the presentation.

Close: Approximately 8:15pm


The Institution of Engineering and Technology

22nd October 2024
7:00pm

Hybrid Event - in person and webinar - registration is required for both

Venue
The Atrium Lecture Theatre, Atrium Building, Waterfront Campus, Ipswich.
https://localevents.theiet.org/bf481c
7:00pm start, teas from 6:30pm

Live Webinar via Teams.
https://localevents.theiet.org/79289b

Anglian Coastal Network

Why software is damaging the environment - and what can be done about it

Chris Watts, Software Engineer and Technologist

How changing software and its development can significantly reduce carbon emissions

Abstract.
According to the UN, the information and communication (ICT) sector including AI, cryptocurrency and datacentres is predicted to generate between 6 and 23% of global carbon emissions by 2030. As software controls the hardware, it contributes indirectly to these emissions. This webinar's aim is to show that most software is highly inefficient and so causes a significant quantity of unnecessary emissions. It also proposes incentives to academia and developer organisations to change their methodologies and tools. The first objective shows the level of software inefficiency by examples. The second, indicates the percentage of global emissions that can be saved by achievable software optimisations. The third proposes an efficiency indicator for marketing purposes to encourage software optimisation. The ability to eliminate up to 11% of global emissions is attainable if we make our software run on average an achievable 10 times faster. Further emissions can be avoided if code and/or data size is diminished and embedded product software is optimised.

About Chris Watts.
Chris started his career designing cutting-edge electronics for minicomputer CPUs and developing very high-performance software. At BT, he led a team developing software for ISDN phones. During 23 years at 3Dlabs, he worked on efficient software development for pioneering 3D graphics chips. He successfully programme managed world-beating chip and software development projects. At Tactiq, Chris managed life-saving medical and automotive embedded software and hardware projects. Though retired, he is working on multiple high-tech and environment projects. He has a BSc (Hons) in Electronics, an MSc in Computer Science, is a Chartered Engineer and a Member of the IET.

Contact
George Williamson
george.williamson@ietvolunteer.org

Book your place at
https://localevents.theiet.org/bf481c

theiet.org/communities

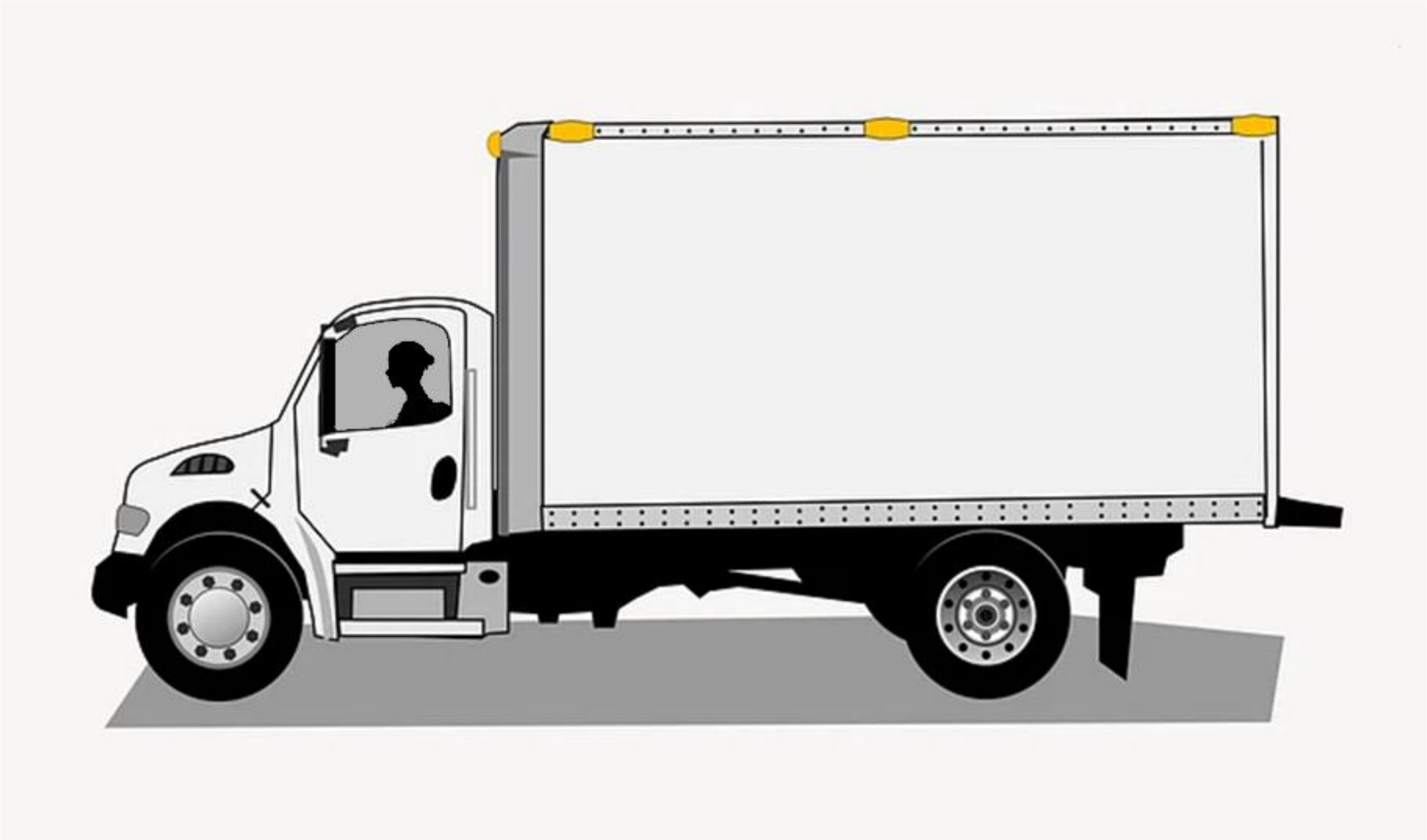
#ForThoseWhoDoMore

2 CPD HOURS

Why software is fuelling climate change – and what can be done about it

IET Webinar by Chris Watts, C.Eng. MSc

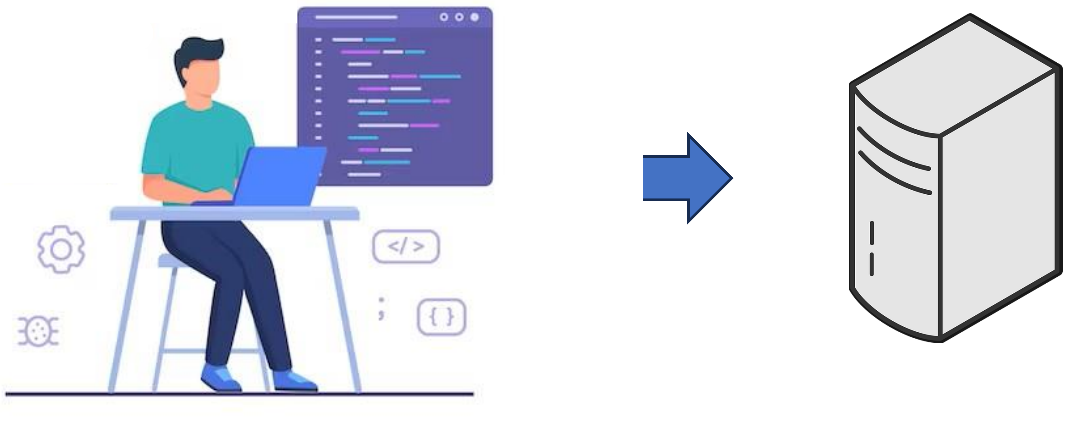
Photo by Matt Palmer

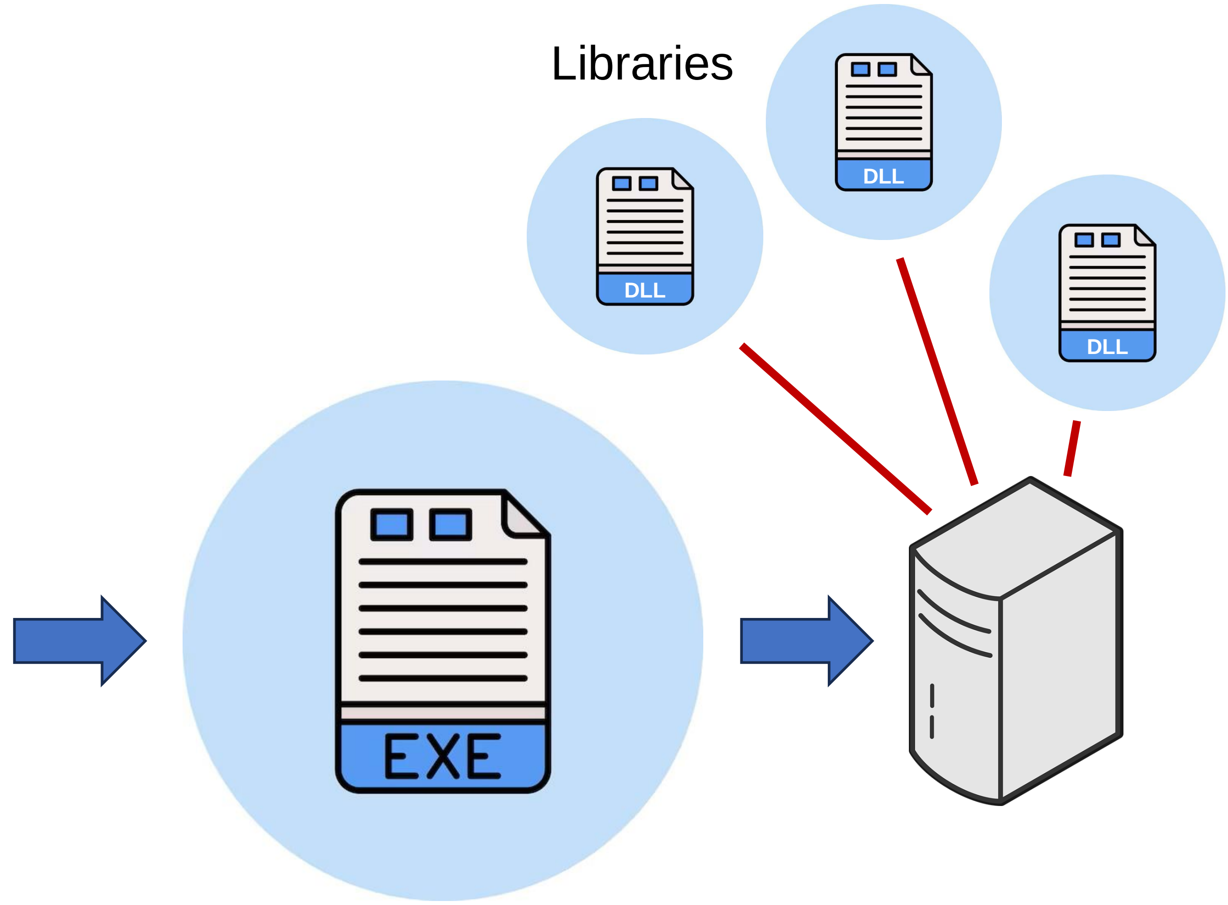


Terminology

Processor	Electronic object, device or part of it that runs software
Carbon emissions	CO2eq = emissions from various greenhouse gases based on global-warming potential
Code	Software

What is Software?



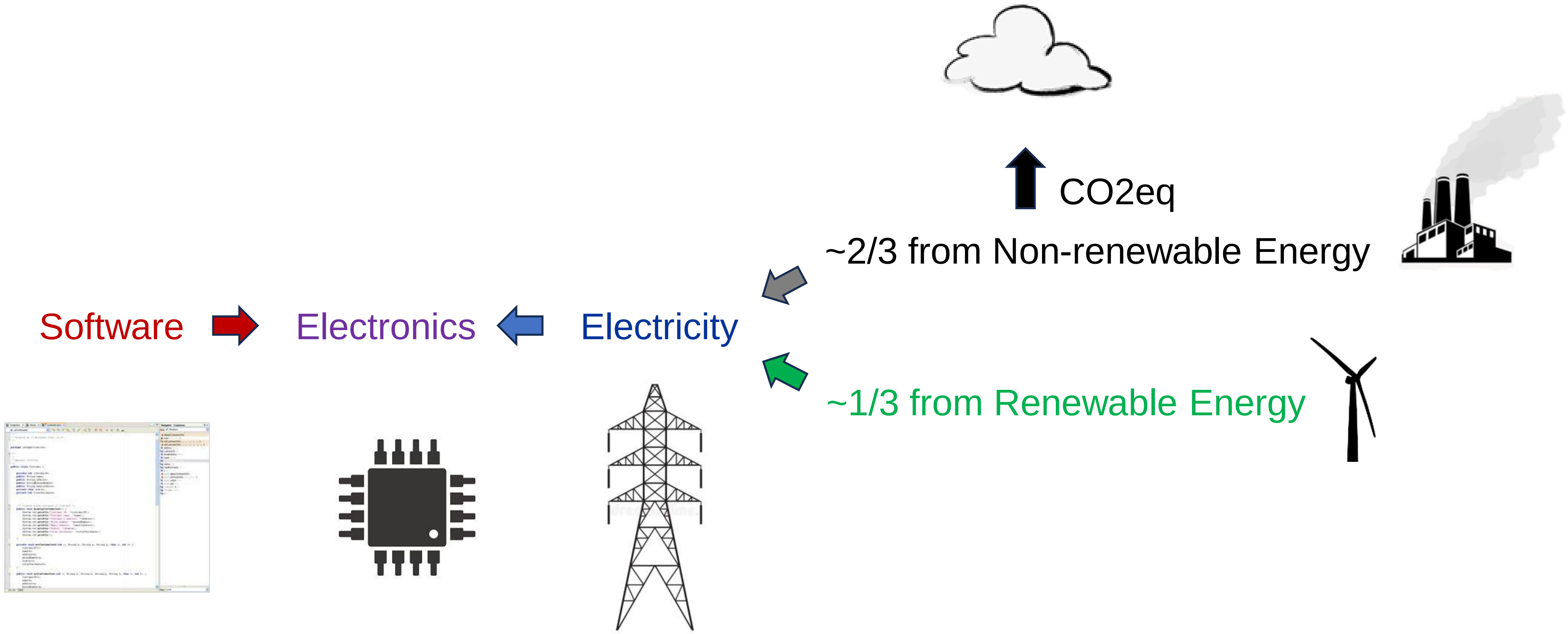


Source Code File(s) Transformed Executable File(s) Run Executable

Excessive Carbon Emissions

Problem 1 –
Inefficient Software Performance

Why does this matter?



Embodied Carbon Example 1

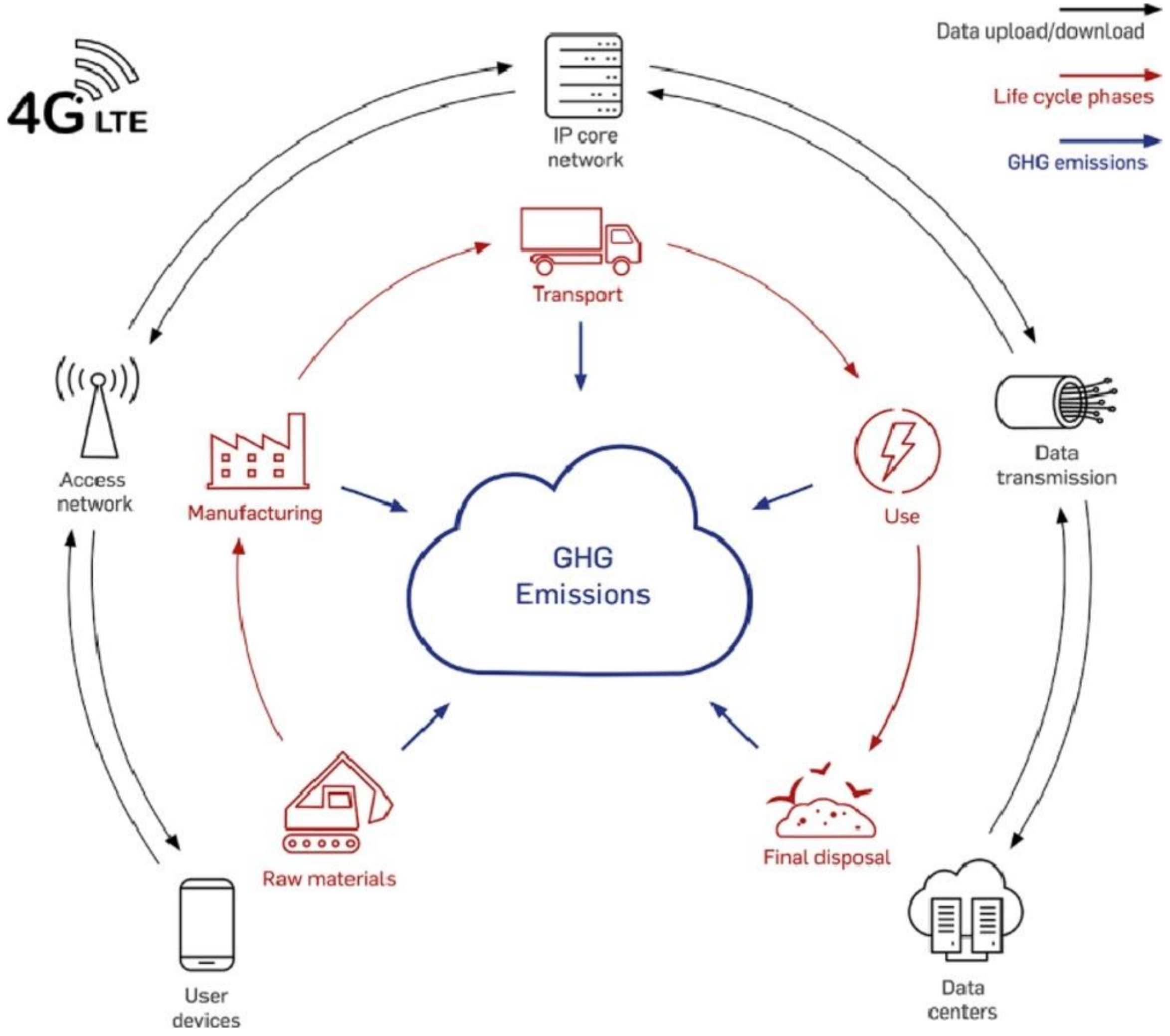


Image from [6]

Embodied Carbon Example 2

Carbon emissions can be attributed to software even if hardware is powered by renewable energy.

Example: Matrix Multiply

4K x 4K

$$
\begin{bmatrix} 7 & 9 & 2 & \dots & 6 \\ 1 & 4 & 8 & \dots & 2 \\ 3 & 7 & 8 & \dots & 2 \\ \dots & \dots & \dots & \dots & \dots \\ 5 & 0 & 6 & \dots & 9 \end{bmatrix}
\text{ X }
\begin{bmatrix} 5 & 9 & 2 & \dots & 0 \\ 4 & 0 & 3 & \dots & 7 \\ 5 & 1 & 8 & \dots & 4 \\ \dots & \dots & \dots & \dots & \dots \\ 2 & 1 & 6 & \dots & 4 \end{bmatrix}
=
\begin{bmatrix} 62 & 25 & 18 & \dots & 33 \\ 17 & 44 & 62 & \dots & 84 \\ 39 & 48 & 64 & \dots & 76 \\ \dots & \dots & \dots & \dots & \dots \\ 92 & 90 & 72 & \dots & 84 \end{bmatrix}
$$

Based on [2]

Example: Matrix Multiply (2)

Version	Implementation
1	Python 2
2	Java
3	C
4	Parallel loops
5	Parallel divide & conquer
6	Add vectorisation
7	Use vector instructions

Based on [2]

Example: Matrix Multiply (3)

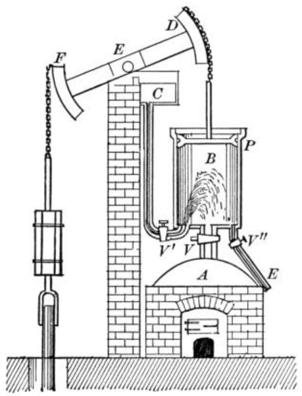
Version	Implementation	Absolute Speedup
1	Python 2	1
2	Java	11
3	C	47
4	Parallel loops	366
5	Parallel divide & conquer	6,727
6	Add vectorisation	23,224
7	Use vector instructions	**62,806**

Based on [2]

Example: Matrix Multiply (4)

Version	Implementation	Absolute Speedup
1	Python 2	1
2	Java	11
3	C	47
4	Parallel loops	366
5	Parallel divide & conquer	6,727
6	Add vectorisation	23,224
7	Use vector instructions	**62,806**

1712 – Newcomen’s steam engine = 0.5% efficient

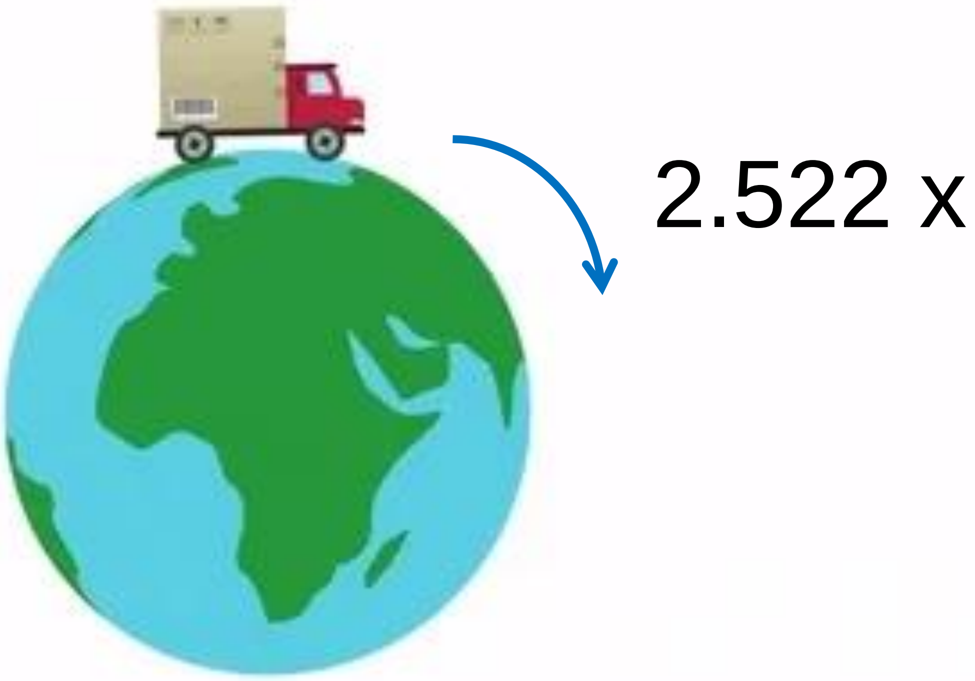


~300X more efficient than Version 1 of software

Based on [2]

If direct route is 1 mile.

Detour comparable to circumnavigating the world 2.5 times.



Python 2 took 7 hours to run the program.
With Python 3 it took **9 hours**.

Original	**Python 2**	**Python 3**
Speed up	**62,806**	**80,750**

Based on [2]

Example: Matrix Multiply (5)

Version	Implementation	Absolute Speedup	Relative Speedup
1	Python 2	1	-
2	Java	11	10.8
3	C	47	4.4
4	Parallel loops	366	7.8
5	Parallel divide & conquer	6,727	18.4
6	Add vectorisation	23,224	3.5
7	Add AVX intrinsics	**62,806**	2.7

Language changes (versions 1–3)

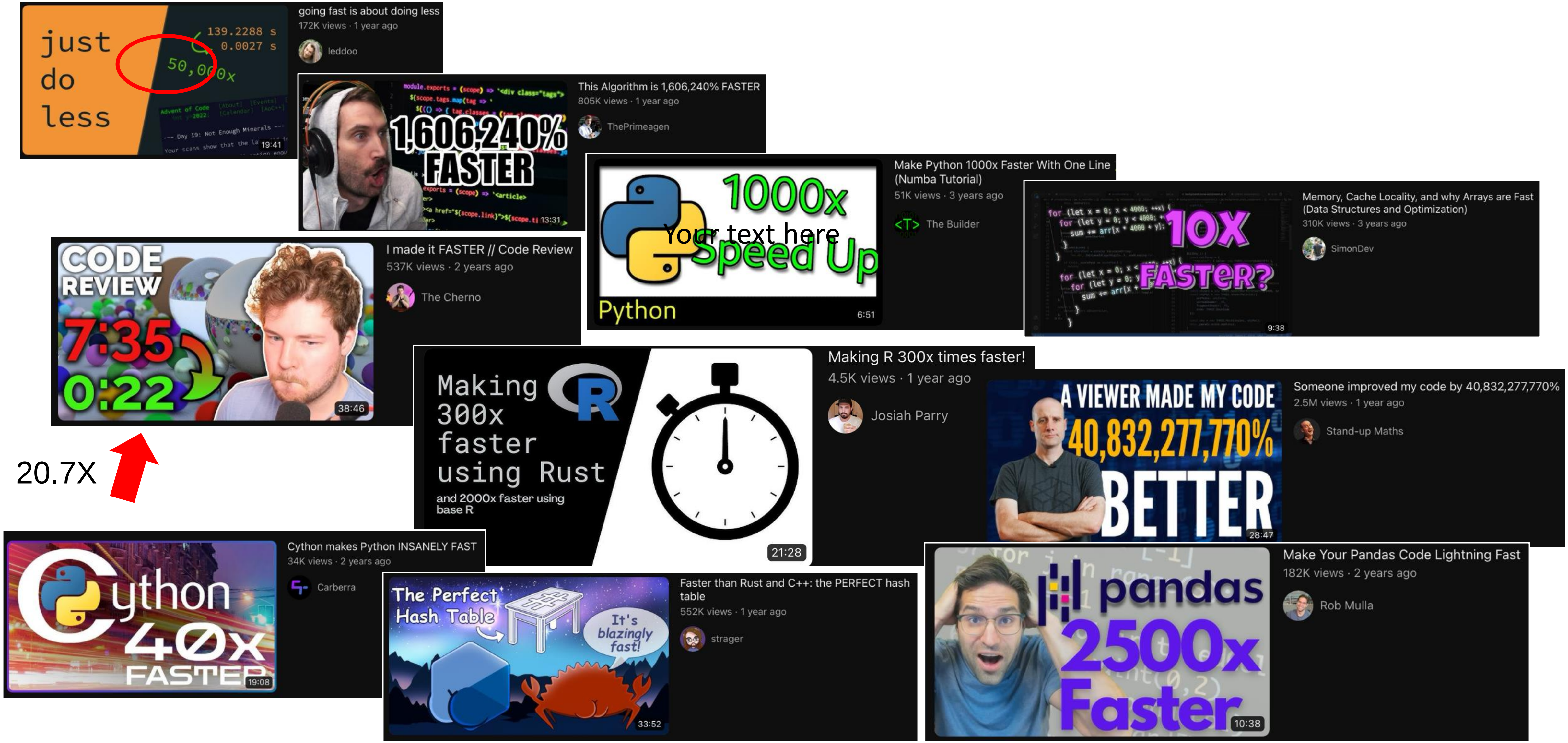
Better use of hardware features (versions 4–5)

Algorithm changes (version 6)

Better use of hardware features (version 7)

Based on [2]

Other examples from Youtube



20.7X

See [3]

What about the Hardware?

HUGI = Hurry Up and Get Idle

From [5]

Programming Languages

- C – fastest
- Python - slow
- C - most energy efficient
- Python - energy inefficient

Table 4
Normalized global results for Energy, Time, and Memory.

Total					
	Energy (J)		Time (ms)		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

From [4]

Software Methodologies - Performance

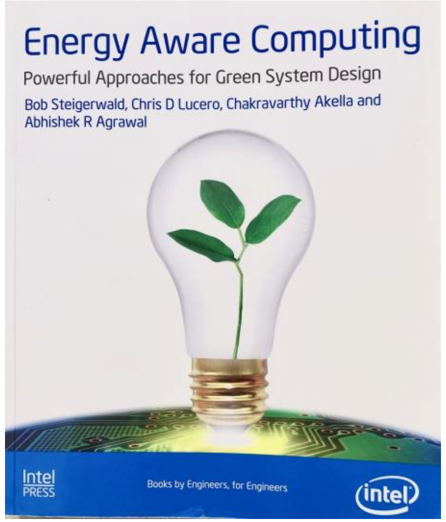
- Procedural vs Object Oriented Programming
  - Some as procedural
  - Some as OOP
- Summary: OOP is slower
- Energy consumption:
  - 14.8% more for OOP

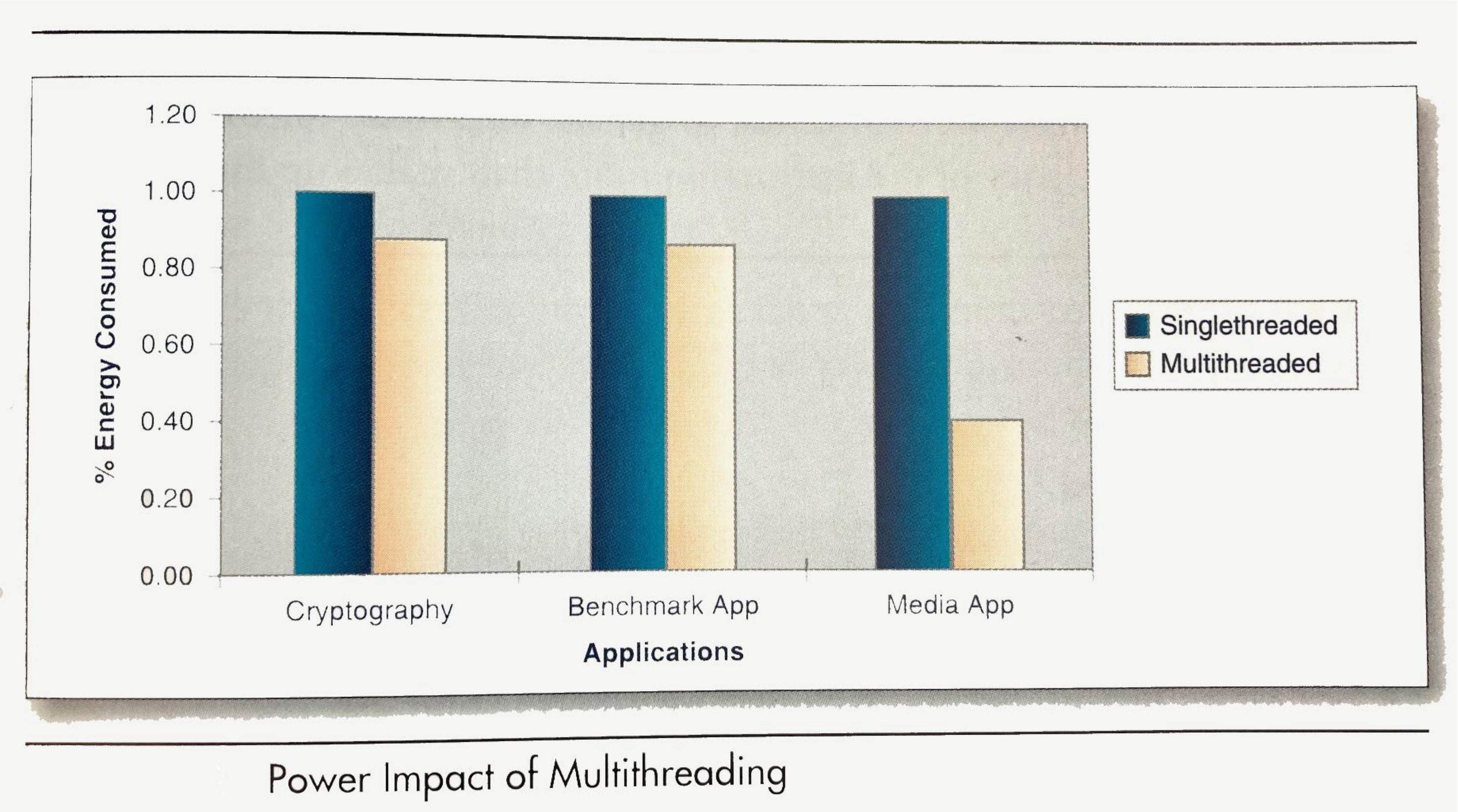
Benchmark	Cycles
OOPACK1_c	77118
OOPACK1_oop	91605
OOP Penalty	*18.79 %*
OOPACK2_c	8303851
OOPACK2_oop	9051974
OOP Penalty	*9.00 %*
OOPACK3_c	635096
OOPACK3_oop	677103
OOP Penalty	*6.61 %*
OOPACK4_c	1606642
OOPACK4_oop	1710665
OOP Penalty	*6.47 %*

From [9]

Object Oriented vs Functional Programming – less clear cut

Using Available Hardware - Multithreading

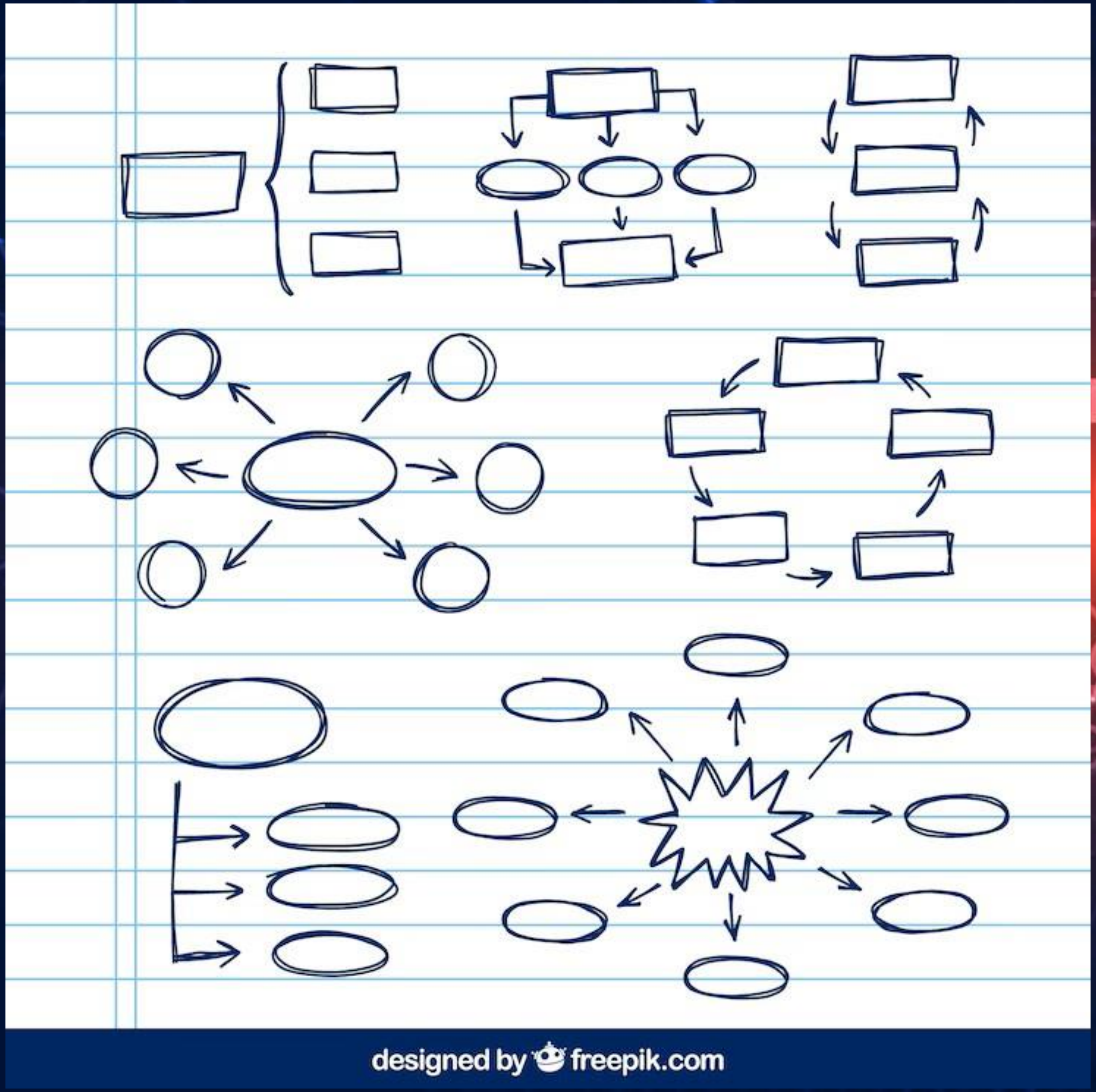




" ... show 12-60% energy savings based on how well they are threaded on a quad-core processor".

From [5]

Algorithm Choice



AI

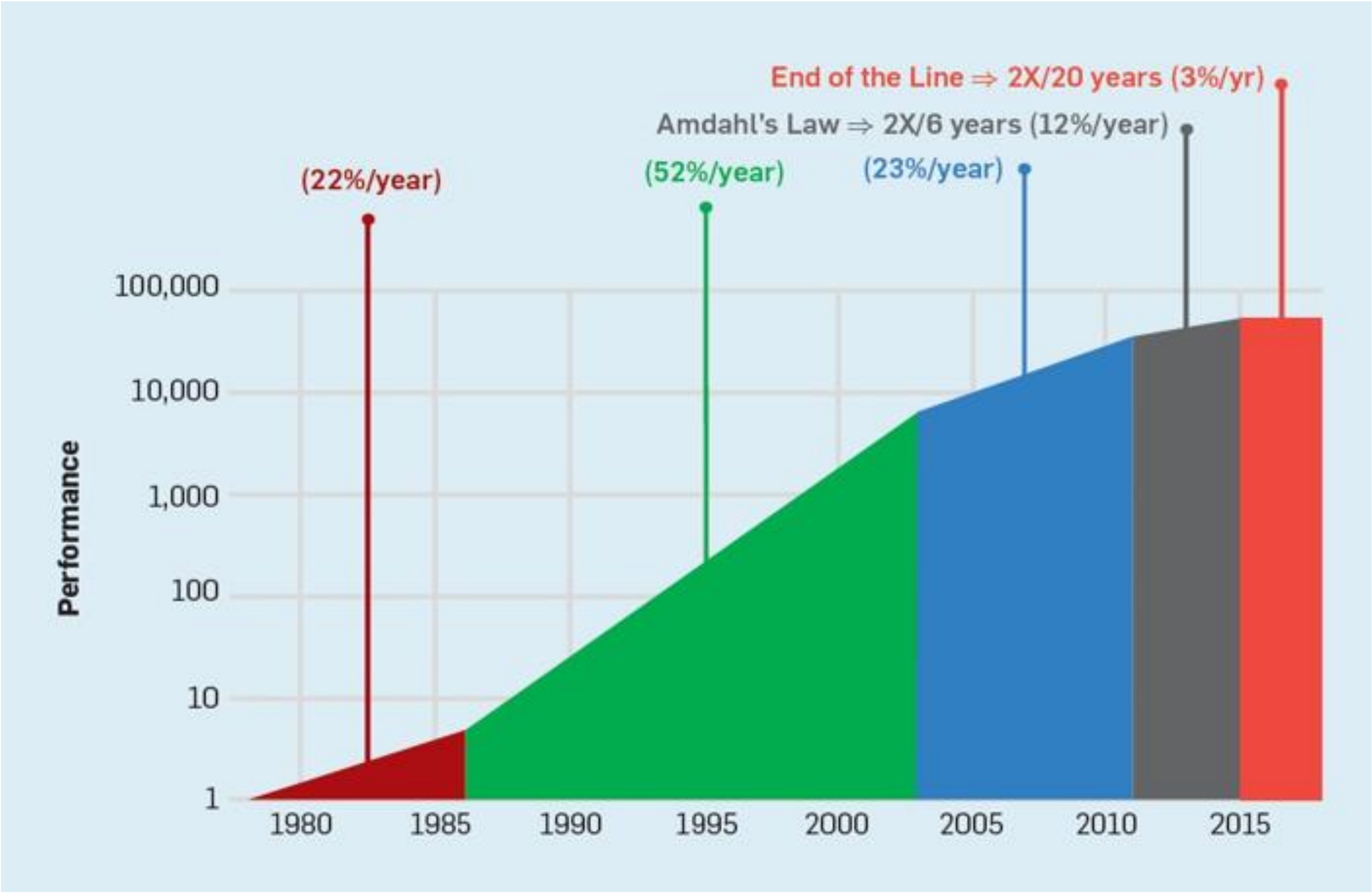
"… it's estimated that Open AI's ChatGPT consumes 2.9 Wh per request"

*"A Generative AI system might use around **33 times** more energy than machines running task-specific software"*

Dr Luccioni and colleagues

From [16] and [12]

Hardware - CPU Integer Performance



"Software is getting slower more rapidly than hardware is becoming faster"

Wirth's/Reiser's Law:

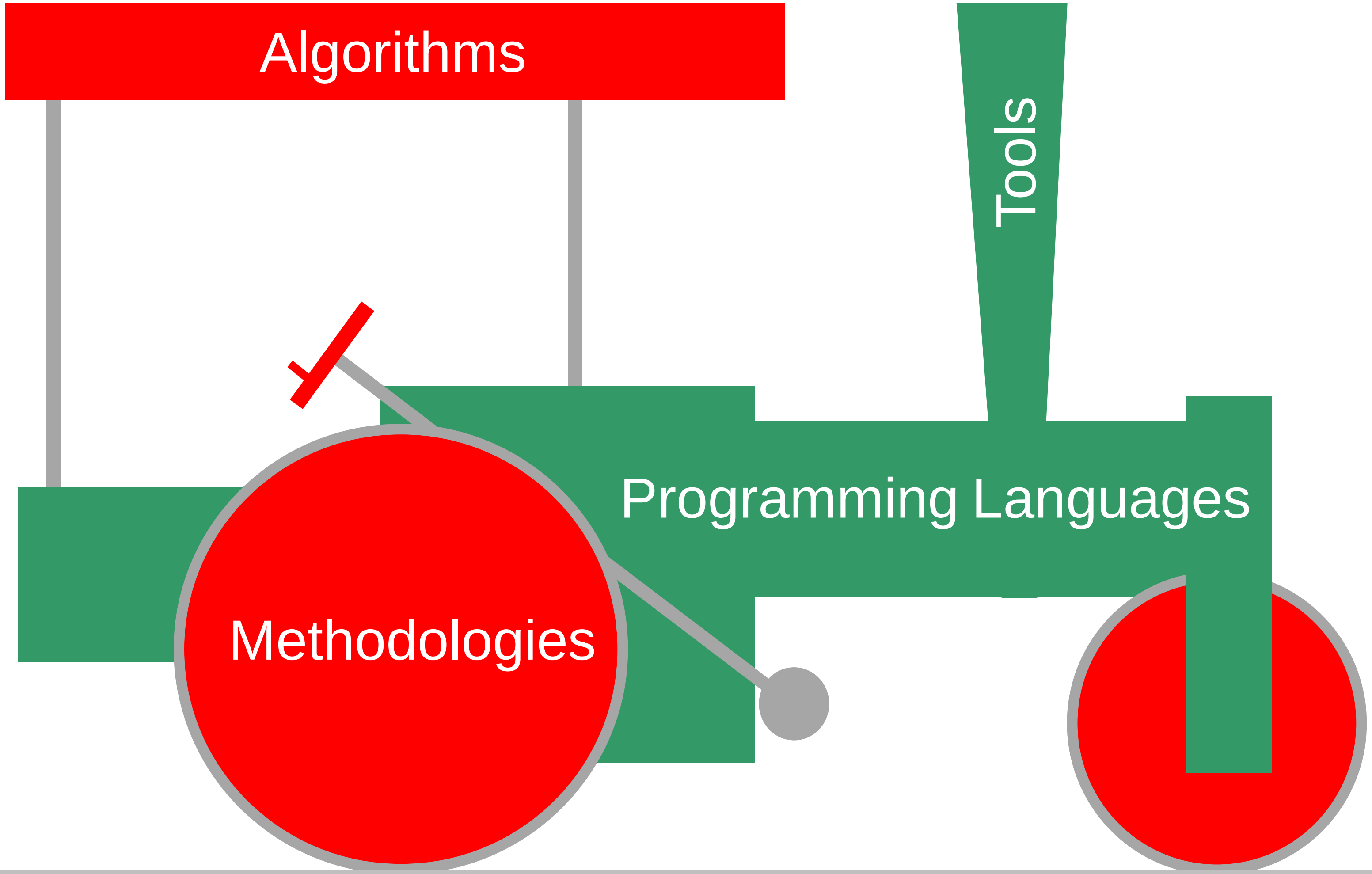
Software can no longer rely on ever faster hardware

From [5]

Exceptions - Performance

The highly efficient software that exists is a tiny minority of the total

Inefficient Software



Result: lower performance, worse user experience, higher energy consumption, higher cost, higher CO2eq emissions

Excessive Carbon Emissions

Problem 1 – Inefficient Performance

Problem 2 – Excessive Code Size

Code Bloat

- Executable code larger in size than necessary.
- Could apply to size of source code
- Niklaus Wirth 1995 article *"A Plea for Lean Software"*

From [15]

Code Bloat – Example 1

Code size comparison of common applications from 2 PCs

Application	March 1999 (KB)	August 2024 (KB)	Size Increase
Database Management	4,569	20,245	4.4 X
Email Client	56	43,093	769.5 X
Spreadsheet	6,985	68,123	9.8 X

"*Code bloat has become astronomical*" by Cliff Harris

- Software tool to upload a file
- Upload tool was:

2,700 files totalling **230MB**

From [8]

Code Bloat - Software Methodologies

- Previous Procedural vs Object Oriented example
- Summary: OO has larger code size

Benchmark	code size (bytes)
OOPACK1_c	180
OOPACK1_oop	212
OOP Penalty	*17.78 %*
OOPACK2_c	308
OOPACK2_oop	424
OOP Penalty	*37.66 %*
OOPACK3_c	260
OOPACK3_oop	356
OOP Penalty	*36.92 %*
OOPACK4_c	620
OOPACK4_oop	804
OOP Penalty	*29.68 %*

From [9]

Excessive Carbon Emissions

Problem 1 – Inefficient Performance

Problem 2 – Excessive Code Size

Problem 3 – Excessive Data

Excessive Data

Effects of Code Bloat or Excessive Data

- Code bloat - Reduced security
- More code or data corruption and greater system downtime
- More RAM memory needed
- More SSD/Disk memory needed
- Moving code or data - More transmission bandwidth consumed
- Greater proportion of embodied carbon attributed to the code and data

Result: lower performance, slower load time, higher energy consumption, higher cost, higher CO_2eq emissions

Benefits of modern software

Why are we in this situation?

Why are we in this situation?

1) Software development priorities:
   - Development time
   - Functionality
   - Usability
   - Code reuse
   - Maintainability
   - Reliability
   - Security
   - Standards
   - Safety (sometimes)

NUMBER 1!

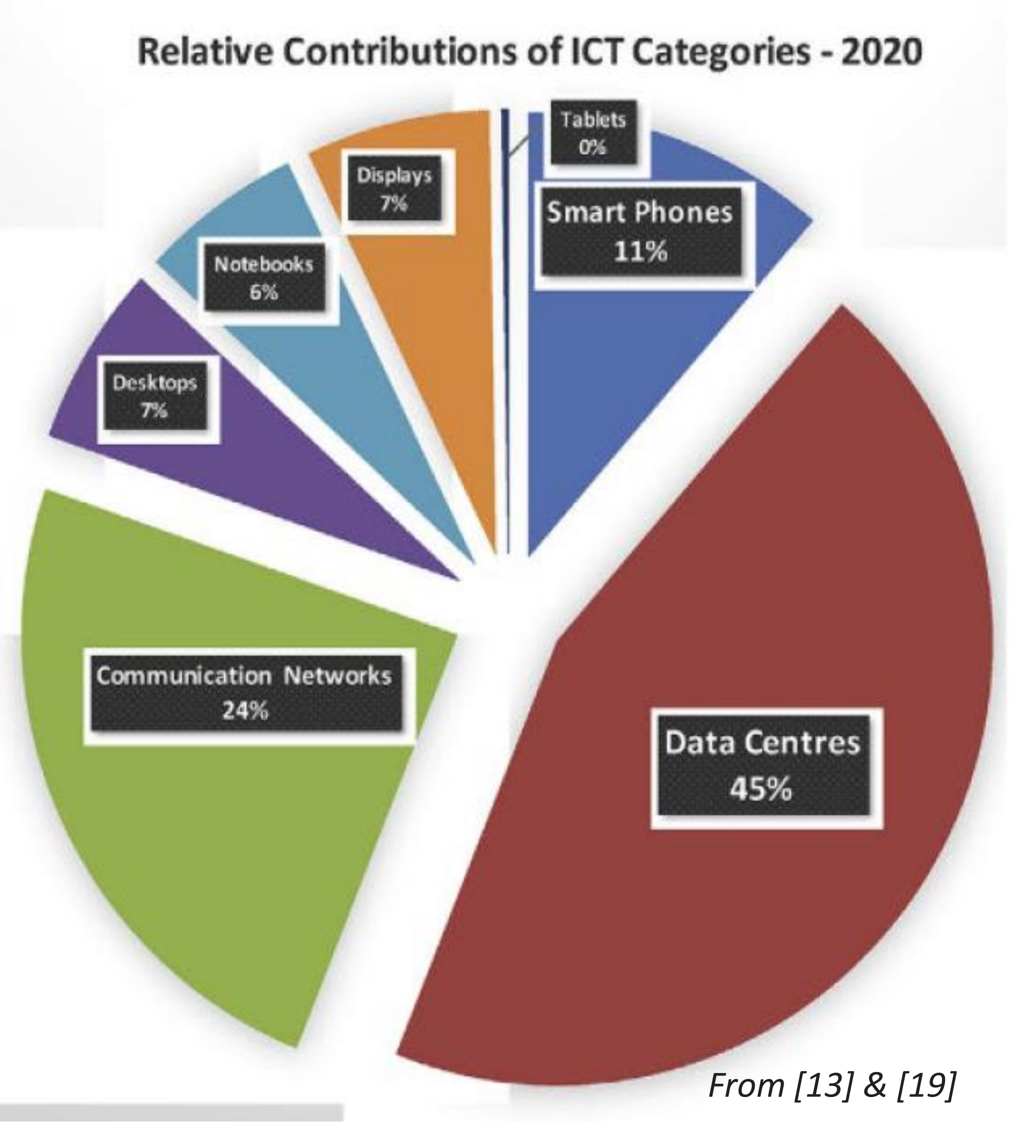
Software performance **X**
Size of code **X**
Size of data **X**

2) No adjustment for hardware improvement slowdown

Why does this matter now?

- Internet, Computing and Telecommunications (ICT) sector
- According to UNEP, by 2030, ICT predicted to account for:

  between **6** and **23%** of global GHG emissions



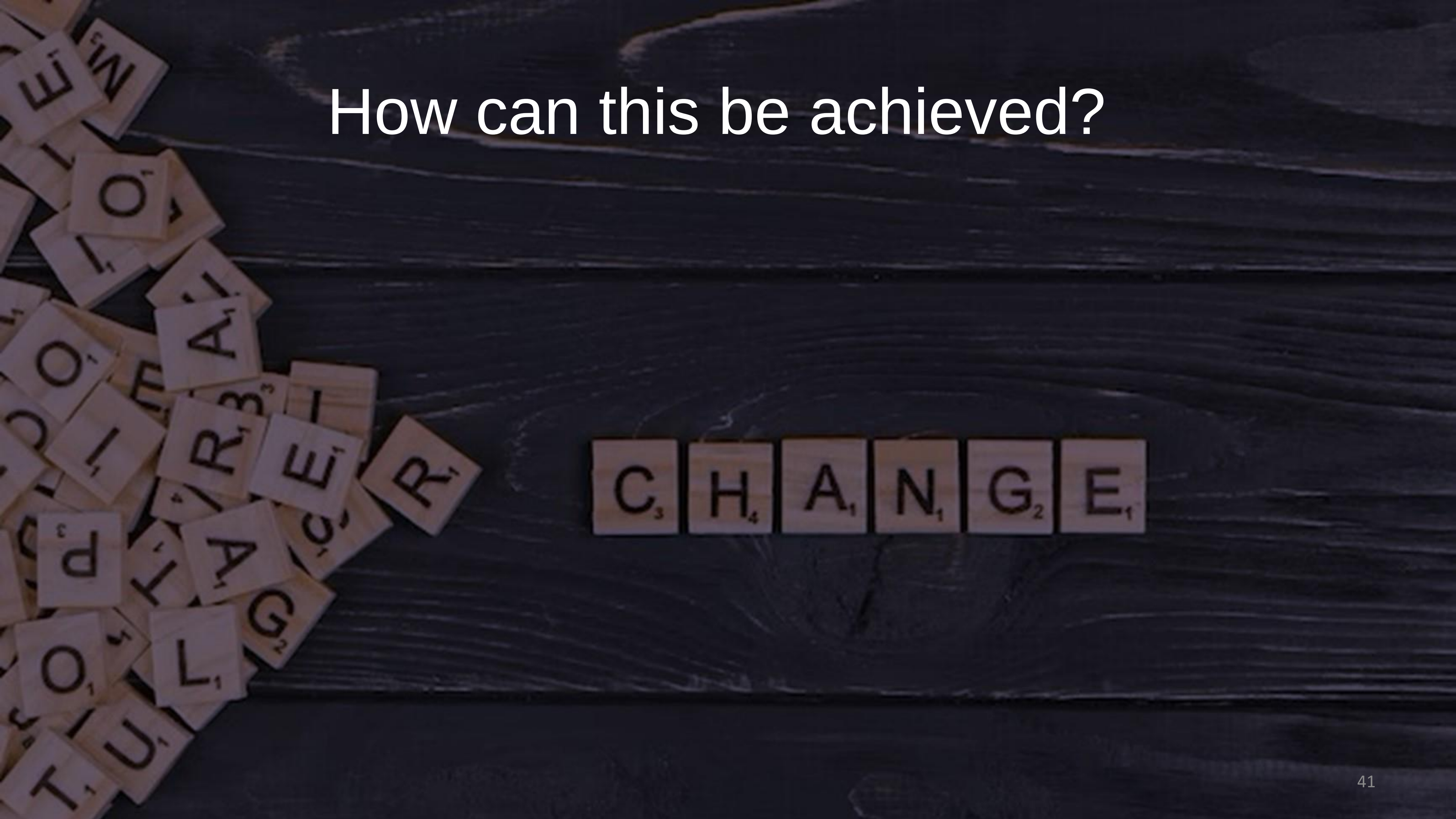
From [13] & [19]

What can be done?

- Change development philosophy
- Optimise software for performance, code and data size

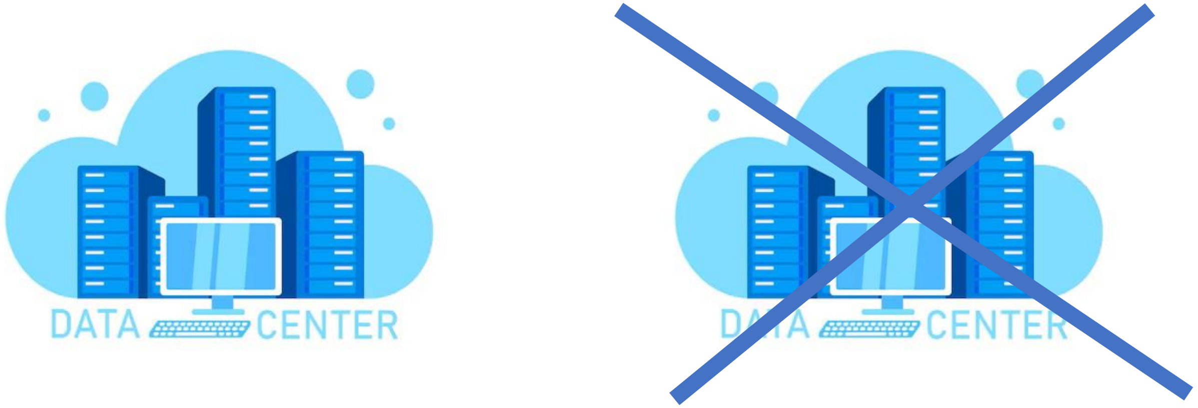
- Aim for **10X s**oftware performance (on average)
  - Vast majority of software can be sped up
  - The speed up in most cases will be >50X

How can this be achieved?

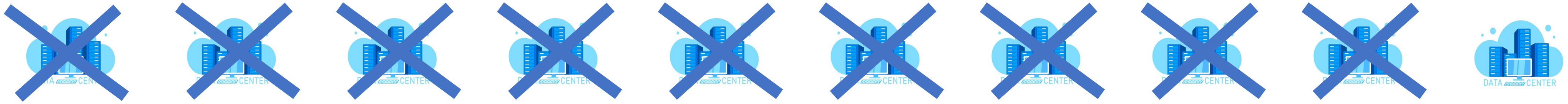


Example - Datacentres

- Double software performance (on average)



- Halves the number of datacentres required
- If 10X improvement, eliminate 9 out of 10 datacentres



Need to optimise software everywhere

Reducing Carbon Emissions

A Solution and an Approach

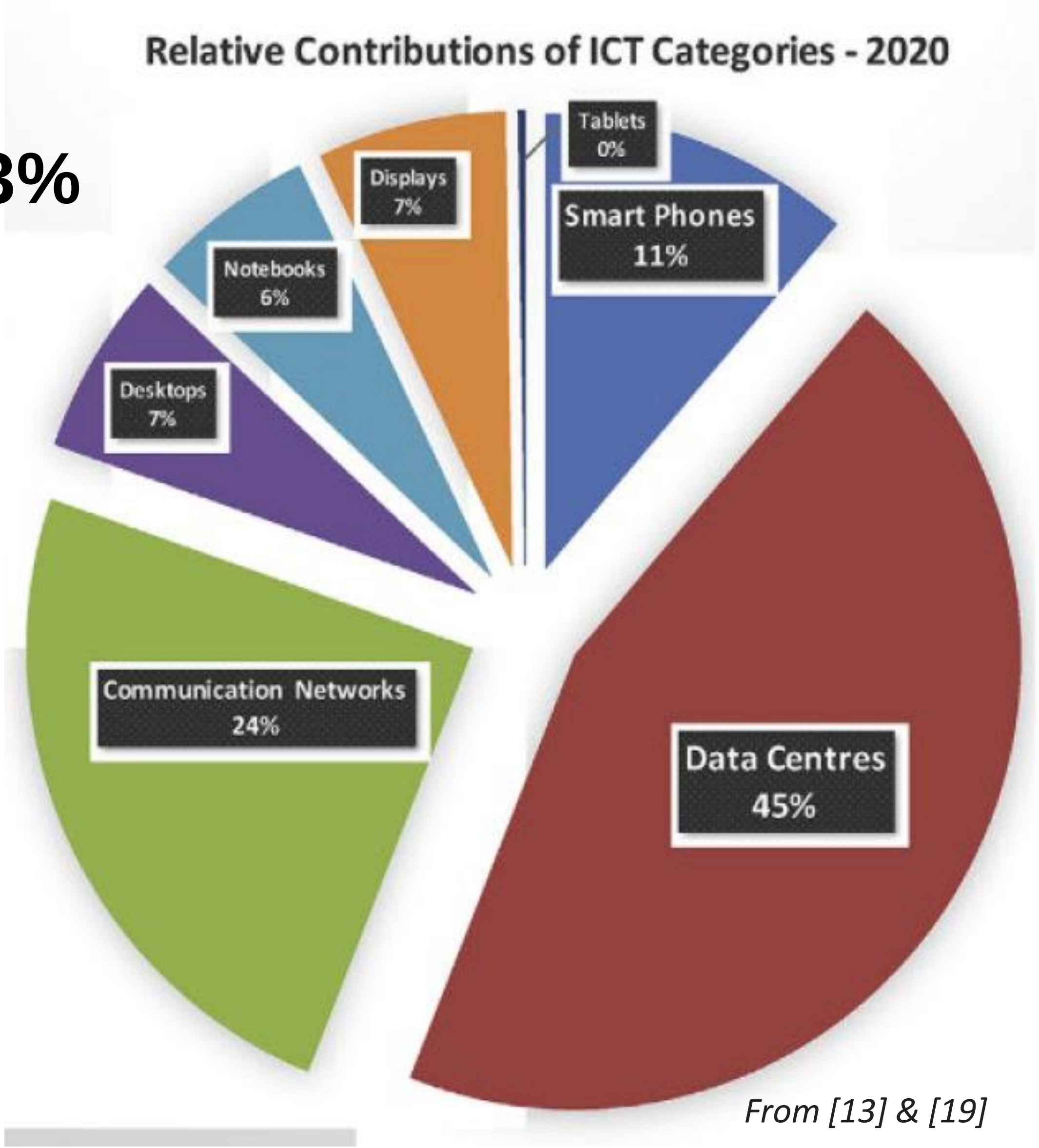
Potential Emissions Savings

- By 2030, global ICT emissions: **6 to 23%**

- Software sped up 10X, eliminates:
  - **90%** of all Datacentre emissions
  - up to **75%** of other ICT emissions

- Global CO2eq emissions due to ICT:

3 to 11%



From [13] & [19]

Additional Emissions Savings

- ICT total CO2eq emissions now: **3 to 11%**
- Plus savings from:
  - Embedded/IOT






  - E&M






  - Embodied carbon
- Plus further savings if:
  - Size of code and/or data is reduced

An Approach

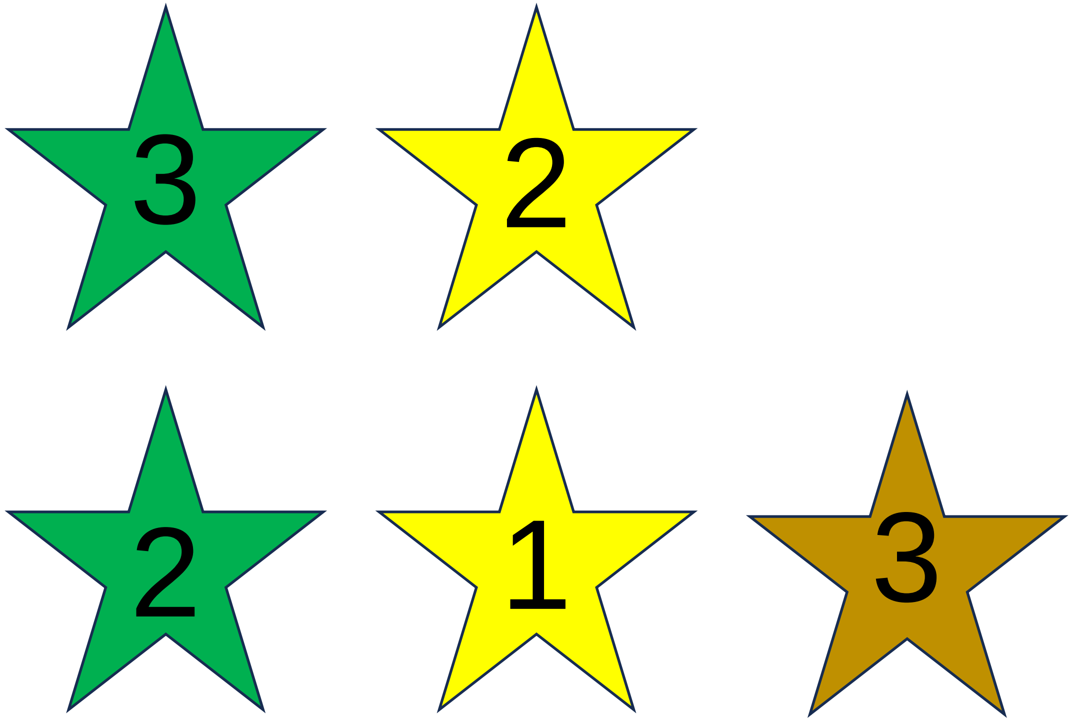
ENERGY STAR

Rating System

Software Stars Example

A word processing application v3.44

A mobile game v2.0




Software
Costing the Earth

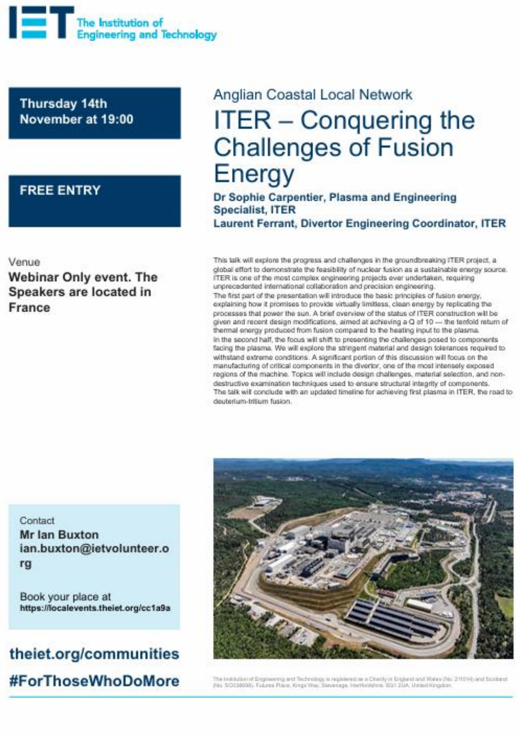

Questions ?
(email: cwatts.greensw@gmail.com)

The Institution of Engineering and Technology

Thank you for your attendance

Future Anglian Coastal Webinars:

- **14th November 2024 - ITER – Conquering the challenges of fusion energy**
- **4th December 2024 – The future is arriving – even in Suffolk**



For more details and how to register please visit:

https://engx.theiet.org/local-networks/ea1

CPD Certificates and Videos for todays and previous Anglian Coastal Network talks will be posted on this site.

References (1)

[1] Wu C. J, Acun B, Raghavendra R, Hazelwood K (2024). *Beyond Efficiency: Scaling AI Sustainably* (IEEE Micro 2024).

[2] Leiserson C. E, Thompson N. C, Emer J. S, Kuszmaul B. C, Lampson B. W, Sanchez D, Schardl T. B (2020). *There's plenty of room at the Top: What will drive computer performance after Moore's law?"* Science 368, eaam9744 (2020) – 5 June 2020

[3] YouTube videos (left to right);

https://www.youtube.com/watch?v=5rb0vvJ7NCY&t=16s&pp=ygUeZ29pbmcgZmFzdCBpcyBhYm91dCBkb2luZyBsZXNz

https://www.youtube.com/watch?v=U16RnpV48KQ&pp=ygUcdGhpcyBhbGdvcml0aG0gaXMgMSw2MDYsMjQwJQ%3D%3D

https://www.youtube.com/watch?v=aNF4DEluWnI&pp=ygUPZmFzdCBpbyBpbiBjKysg

https://www.youtube.com/watch?v=OiMZtjSZVOw&pp=ygURbWFrZSBweXRob24gMTAwMHg%3D

https://www.youtube.com/watch?v=247cXLkYt2M&pp=ygUWbWVtb3J5LCBjYWNoZSBsb2NhbGl0eQ%3D%3D

https://www.youtube.com/watch?v=mOSirVeP5lo&t=1123s&pp=ygUfSSBtYWRlIGl0IEZBU1RFUiAvLyBDb2RlIHJldmlldw%3D%3D

https://www.youtube.com/watch?v=-v9qaqaj4Ug&pp=ygUNTWFraW5nIFlgMzAweA%3D%3D

https://www.youtube.com/watch?v=c33AZBnRHks&t=121s&pp=ygUfc29tZW9uZSBpbXByb3ZlIG15IGNvZGUgYnkgNDA4IA%3D%3D

References (2)

https://www.youtube.com/watch?v=ODnHNd5xcNg&pp=ygUTQ3l0aG9uIG1ha2VzIFB5dGhvbg%3D%3D

https://www.youtube.com/watch?v=DMQ_HcNSOAI&pp=ygUYRmFzdGVyIHRoYW4gUnVzdCBhbmQgQysr

https://www.youtube.com/watch?v=SAFmrTnEHLg&pp=ygUVTWFrZSB5b3VyIFBhbmRhcyBDb2Rl

[4] Pereira R, Couto M, Ribeiro F, Rua R, Cunha J, Paulo Fernandes J, Saraiva J (2021) *Ranking programming languages by energy efficiency,* Science of Computer Programming 205 (2021) 102609

[5] Hennessy J. L, Patterson D. A (2019) *A New Golden Age for Computer Architecture*, The ACM, February 2019, Vol. 62, No. 2

[6] Ruiz D, San Miguel G, Rojo J, Teriús-Padrón J. G, Gaeta E, Arredondo M. T, Hernández J. F, Pérez J (2022) *Life cycle inventory and carbon footprint assessment of wireless ICT networks for six demographic areas,* Resources, Conservation & Recycling 176 (2022) 105951

[7] Potdar A. M, Narayan D. G, Kengond S, Mulla M. M (2020) *Performance Evaluation of Docker Container and Virtual Machine,* Procedia Computer Science 171 (2020) 1419–1428

[8] Harris C (2022) *Code bloat has become astronomical*, https://www.positech.co.uk/cliffsblog/2022/06/05/code-bloat-has-become-astronomical/, Cliffski's Blog, (2022)

[9] Chatzigeorgiou A, Stephanides G (2002), *Evaluating Performance and Power of Object-Oriented Vs. Procedural Programming*, J. Blieberger and A. Strohmeier (Eds.): Ada-Europe 2002, LNCS 2361, pp. 65-75

in Embedded Processors

[10] Alic D, Omanovic S, Giedrimas (2016) *Comparative analysis of functional and object-oriented programming*, MIPRO (2016)

References (3)

[11] Baraniuk C (2024) *Electricity grids creak as AI demands soar,* BBC News article, https://www.bbc.co.uk/news/articles/cj5ll89dy2mo

[12] Luccioni A. S, Jernite Y, Strubell E (2024) *Power Hungry Processing: Watts Driving the Cost of AI Deployment?* FAccT '24, June 03–06, 2024

[13] (2020) *Greenhouse gas emissions in the ICT sector - Trends and methodologies,* UNEP DTU, https://c2e2.unepccc.org/wp-content/uploads/sites/3/2020/03/greenhouse-gas-emissions-in-the-ict-sector.pdf

[14] Steigerwald R, Lucero C. D, Akella C, Agrawal A. R (2011) *Energy Aware Computing*, Intel Press ISBN 13 978-1-934053-41-6

[15] Wirth N (1995) *A Plea for Lean Software*, Computer IEEE 1995

[16] IEEE Computer Society Team (2024) *Does the Promise of Artificial Intelligence Outweigh Its Environmental Impact?* IEEE Computer Society 26 September 2024 https://www.computer.org/publications/tech-news/research/artificial-intelligence-environmental-impact

[17] Gupta U, Young G K, Lee S, Tse J, Lee H-H S, Wei G-Y, Brooks D, Wu C-J (2020) *Chasing Carbon: The Elusive Environmental Footprint of Computing,* 28 Oct 2020 https://arxiv.org/pdf/2011.02839

[18] Malmodin J (2015) *Exploring the effect of ICT solutions on GHG emissions in 2030*, EnviroInfo 2015, https://www.atlantis-press.com/proceedings/ict4s-env-15/25836149

[19] Andrae A S G, Edler T (2015) On Global Electricity Usage of Communication Technology Trends to 2030, Huawei Technologies Sweden AB, Challenges 2015 https://www.mdpi.com/2078-1547/6/1/117.

References (4)

[] Baraniuk C (2024) *Electricity grids creak as AI demands soar,* BBC News article, https://www.bbc.co.uk/news/articles/cj5ll89dy2mo

[12] Luccioni A. S, Jernite Y, Strubell E (2024) *Power Hungry Processing: Watts Driving the Cost of AI*

Deployment? FAccT '24, June 03–06, 2024

[13] (2020) *Greenhouse gas emissions in the ICT sector - Trends and methodologies,* UNEP DTU, https://c2e2.unepccc.org/wp-content/uploads/sites/3/2020/03/greenhouse-gas-emissions-in-the-ict-sector.pdf

[14] Steigerwald R, Lucero C. D, Akella C, Agrawal A. R (2011) *Energy Aware Computing*, Intel Press ISBN 13 978-1-934053-41-6

[15] Wirth N (1995) *A Plea for Lean Software*, Computer IEEE 1995

[16] IEEE Computer Society Team (2024) *Does the Promise of Artificial Intelligence Outweigh Its Environmental Impact?* IEEE Computer Society 26 September 2024 https://www.computer.org/publications/tech-news/research/artificial-intelligence-environmental-impact

[17] Gupta U, Young G K, Lee S, Tse J, Lee H-H S, Wei G-Y, Brooks D, Wu C-J (2020) *Chasing Carbon: The Elusive Environmental Footprint of Computing,* 28 Oct 2020 https://arxiv.org/pdf/2011.02839

[18] Malmodin J (2015) *Exploring the effect of ICT solutions on GHG emissions in 2030*, EnviroInfo 2015, https://www.atlantis-press.com/proceedings/ict4s-env-15/25836149

[19] Andrae A S G, Edler T (2015) On Global Electricity Usage of Communication Technology Trends to 2030, Huawei Technologies Sweden AB, Challenges 2015 https://www.mdpi.com/2078-1547/6/1/117.